Supplementary document

2022

— S1 —

1 Single omics methods

Due to the large amount of data to be exploited in a multi-omics framework, a unique pre-processing at each multi-omics layer has been performed. This sometimes included the imputation of missing values as well as the introduction of methods to reduce the dataset dimensions to make judicious selections of omics variables. A total of 444 participants were included in the study but only 330 of them had all the metabolomic, transcriptomic, clinical, PRS and methylation data. In order to gain statistical power in the pre-processing stages, inequal numbers of participants were used to make variable selections (Figure 1, original paper). Thus, the pre-processing of each layer was performed using the maximum number of individuals from the initial 444 participants. Once the 330 participants were selected in the multi-omics phase, only the twin pairs were selected for the modeling phase. The remaining 20 singletons were used to tune the sMBPLS design (see original paper).

1.1 Blood pressure measurements

Four systolic and diastolic blood pressure measurements were collected for n=444 participants with no missing blood values. A blood pressure correction was performed for participants taking antihypertensive drugs corresponding to an increase of 15mmHg and 10mmHg in systolic and diastolic measurements respectively. Due to stress or white coat effect, possible biases were suspected (Vilaplana, 2006). The use of all these systolic and diastolic blood measurements in an averaged form was therefore questioned. Two multiple linear regressions confirmed these doubts by showing a significantly non-null association of the measurement times once one-hot encoded on both diastolic and systolic blood pressure (two-tailed Student's t test, $p - value < 5\%$) controlling for sex, age and zygosity status of the participant. An approach based on euclidean distances between measurements then highlighted the proximity between two groups of measurements; the first two and the last two. The choice was therefore made to average only the last two diastolic (resp. systolic) measurements since they differed strongly from the two first measurements, which corresponds to the empirical choice made in Huang et al (Huang et al., 2018). Hereafter, the term diastolic (resp. systolic) blood pressure (DBP) (resp. SBP) refers to the diastolic (resp. systolic) variable by means of the last two measurements.

1.2 Metabolomics data

Metabolomic data were collected for n = 434 individuals, including 212 twin pairs and 10 singletons. A total of 234 numerical variables were subject to pretreatment. The variable measuring peroxide levels was only measured in 64 of the 434 individuals and was therefore removed from the set of variables. Of the remaining 233 variables, 44 had at least one missing value but only two had a missing value rate of more than 10%. Regarding the nature of these missing values, it was judged consistent to consider them as missing (completly) at random (MAR/MCAR) since the presence of missing values is only subject to measurement errors. Before imputing missing values, a sorting of the variables according to the inter-variable correlations was carried out within the framework of a complete case, i.e. by considering only the 296 individuals with no missing values. Variables for which at least one other variable correlated at more than 90% in absolute value was detected were removed, taking care to keep at least one representative among each batch of correlated variables. Variables with missing values have been deleted in priority, thus reducing the number of variables to be imputed to 29 variables instead of the initial 44.

While the use of multiple imputation methods is now globally accepted as a gold standard, the use of these methods requires a rigorous methodology (Hayati Rezvan et al., 2015). Due to persistent collinearities within the pre-selected variables, the use of the *Amelia* (Honaker et al., 2011) package was preferred to the mice (van Buuren et al., 2011) package as it allows the integration of a Ridge penalty to counter the problems of matrix inversion caused by the presence of persistent correlations. This penalization reduces the variance at the cost of a slight bias expected as to be minimal. Amelia was therefore used to impute missing values from the Expectation-Maximization with Bootstrapping algorithm (EMB). Imputation assumes as main assumption that the complete data, both observed and unobserved, are multivariate normal. This central assumption was considered to be verified as the metabolomic variables are expected to follow gaussian distributions. The Ridge penalty was set at $0.5\%n$ and did not exceed the $1\%n$ threshold as advised by the authors of the package. A total of m = 50 datasets were generated in order to calculate pooled estimates. A minimum burn-in of 100 iterations has also been stated. Finally, the distributions of imputed values versus non-imputed values were performed as a diagnostic (Abayomi et al., 2005) and only one variable (LVLDLFC: free cholesterol to total lipids ratio in large VLDL) was removed after diagnosis due to aberrant distributions, leading to a final dataset composed of 105 metabolomic variables.

1.3 Methylation data

A total of 360 samples corresponding to 360 participants for whom blood measurements were available were obtained via Illumina 450k. The use of the *minfi* (Aryee et al., 2014) package allowed pre-processing of the raw data. Quality control was performed and no samples were rejected by observing the chipwide medians of the Meth and Unmeth channels (classic cutoff of 10.5). According to Cazaly et al (Cazaly et al., 2016), stratified quantile normalization (sometimes combined with *ComBat* (Leek et al., 2012)) were consistently found to be the most appropriate normalization method in Illumina 450k methylation array data in familial analyses. Stratification by quantile was therefore carried out via *minfi* but no correction for the batch effect was made as the batch sizes were considered too small. The sex of the participants was recorded via clinical data and as the presence of SNPs in the probe body or at the nucleotide extension can have important consequences on the downstream analysis, probes with high bias potential were removed via the minfi package as advised by its authors. A total of 11'648 probes were also removed from the study because they were affiliated with the X and Y sex chromosomes. Finally, all values were transformed into β-values, i.e., defined as the ratio between the methylated intensity of the signal and the total intensity (methylated and unmethylated with one constant usually set to 100).

A selection of the probes once annotated by their genes was made via elastic-net (Zou et al., 2005). The elastic- net regression is defined as a penalised regression integrating a convexity coefficient $\alpha \in [0, 1]$ allowing the combination of Lasso penalisation ($\alpha = 1$) and Ridge penalisation ($\alpha = 0$). This method has many advantages over a Lasso or Ridge regression, making it a commonly used method in exploratory approaches (Domingo-Relloso et al., 2021; Benton et al., 2017) in the framework of correlated epigenetic variables. It allows the selection of groups of variables, which a traditional Lasso regression cannot do. It also allows the selection of a larger number of variables than a Lasso regression for which the limit of selected variables is majorized by the number of individuals ("$p < n$"), which in the context of epigenetic variables is restrictive (Waldmann et al., 2013). In order to reinforce its use, a multi-task design re- gressing a pair of variables in a multi-Gaussian framework was carried out via the use of *glmnet* (Friedman et al., 2010).

It has been judged important to correct for certain effects such as age and sex, which are known to be de facto linked to methylation (Boks et al., 2009). It was considered heavy-handed to include clinical control variables in the modeling within epigenetic variables, as their predictive potential was too strong and can have a huge impact on the norm of the coefficient matrix to be penalized. It was therefore decided to linearly regress systolic and diastolic blood pressure by age, sex and body mass index (BMI) in order to solve the final model on the systolic and diastolic residuals (SBPr,DBPr) obtained, i.e. on the part of the information not explained by these control variables. Due to the existence of an outlier (BMI:36 kg.m2, SBP: 230.2 mmHg, DBP: 116.5 mmHg, Age: 68 years old, Male) who was too important in the sense of Cook's distance, the control was performed without taking him into account in order to improve the quality of the control regression. This participant was included in the downstream study with a predicted residual systolic and diastolic pressure. The classical assumptions of linear regression such as homogeneity or normality of the residuals have been checked graphically. Finally, the convexity parameter α was set 0.15 i.e. close from 0.10 as recommended by Waldmann et al. (Waldmann et al., 2013). The λ coefficient penalizing the norm of the coefficient matrix was tuned from a 10-fold cross validation minimising the Mean Square Error (MSE) criterion. This procedure led to the selection of 545 CpG sites. A representation in the first PCA plane supported this selection insofar as the individuals were arranged in the reduced space according to the non-residual diastolic/systolic measurements.

1.4 Transcriptomic data

RNA samples were extracted from peripheral leukocytes using the miRNeasy Mini Kit. Gene expression data at the transcriptome level was obtained using a gene expression chip. This chip targeted over 48,000 probes that provide transcriptomic coverage of well-characterised genes (see Huang et al (Huang et al., 2018) for details of how the data was collected). Of the 402 peripheral blood samples, high quality RNA was obtained from 391 subjects. Of these 391 participants, two did not participate in the selection of transcriptomic variables because their blood records were discarded at the beginning of the study (see subsection 1.1). The use of the *lumi* (Du et al., 2008) package allowed

the first steps of the data processing to be carried out. The addition of control data was carried out. No background correction was used because the data was not previously corrected when using beadArray. A Variance-Stabilizing Transformation (VST) was performed instead of a log_2 transformation because it allows for better correction of weak signals (Lin et al., 2008). A Robust Spline Normalization (RSN) combining quantile and loess normalization was performed iteratively using a random participant as reference. Only probes with a p-value of detection of less than 5% in at least half of the participants were retained; a total of slighty more than 19'000 probes were therefore kept. Probes present on the sex chromosomes were also removed before analysis.
As was done in the pre-processing of methylation data (see section 1.3), it was preferred to linearly regress SBP and DBP measurements before selecting variables by control variables to avoid detecting too many probes unrelated to the phenotype of interest. A control on sex and BMI was performed beforehand in order to build residual diastolic and systolic measures corresponding to the residuals of the linear regression performed. The presence of an outlier, the same person as above in section 1.3. detected via Cook's distance justified the regression without considering him, before reintegrating this participant into the analysis with predicted residual diastolic and systolic values. A Partial Least Square regression in a sparse version implemented in the mixOmics (Rohart et al., 2017) package was then run to simultaneously regress the DBPr-SBPr pair in a matrix framework. Two components were selected by 10-fold cross-validation repeated 50 times using the mean square error of prediction (MSEP) as the optimization criterion. A total of 300 probes to be selected and equally distributed over the two components were chosen before the procedure (argument keep.X). In order to keep only the most consistant probes, a stability criterion was used. This consisted of counting the number of times a probe appeared over the course of the repetitions by cross-validation. Only probes with a stability greater than 0.8 were retained, i.e., only probes selected in at least 80% of the iterations during the cross-validation procedure were kept. A total of 83 probes were then retained, of which two were discarded because they did not point to any known gene.

Among these 81 probes, we note the presence of two pairs of probes pointing to the *MYADM* (*MYADM* and *MYADM.1*) and *CD97* (*CD97* and *CD97.2*) genes. This further demonstrates the ability of sparse Partial Least Square to select variables in the framework of correlated data. Among the 10 most stable probes in the selection, we notice the presence of *CD97* and *MYADM* as well as *TIPARP* and *TPPP3* which were all replicated (Zeller et al., 2017). The presence of the *MOK* (*RAGE*) probe among the ten most stable probes is noteworthy as it has been shown via mixed models on the same cohort that MOK was associated with blood pressure (Huang et al., 2018).

1.5 Clinical PRS data

A total of 25 clinical variables including sex , BMI and lymphocyte counts were collected for 440 participants. Among these 25 variables, two were built from raw clinical variables: BMI and mean pulse rate. Imputation of missing values was only performed on 3 individuals for whom all omics layers were available. The imputation of these participants was carried out via the mice package using the random forest method, which is well suited to imputation in the context of chained equations from mixed categorical and quantitative data. A total of m = 50 datasets were generated over 30 iterations before calculating the pooled estimates. Graphical representations and convergence diagnostics were investigated to ensure the quality of the imputation. Because alcohol consumption is known to be related to blood pressure (Salvador et al., 2019; Keil et al., 1991), a variable named Alc characterizing consumption in grams per month (Kaprio et al., 1987) was added to the set of clinical variables.

Four Polygenic Risk Scores (PRS) were also added to the 330 participants for whom all omics layers were available. Two of these PRS are associated with diastolic and systolic blood pressure respectively (UK Biobank: http://www.nealelab.is/uk-biobank/). The other two PRS characterise the risk of coronary artery disease (Nikpay et al., 2015) and the risk of obesity (Yengo et al., 2018). These polygenic variables were z-score transformed and PCA corrected (Price et al., 2006) in order to be exploited in the multi-omic phase.

— S2 —

YFS Methylation pre-processing

DNA was extracted from EDTA blood samples collected during the 2011 follow-up using a Wizard Genomic DNA Purification Kit (Promega Corporation, Madison, WI, USA) according to the manufacturer's instructions. Genome-wide DNA methylation levels were obtained using Illumina Infinium MethylationEPIC BeadChips, following the protocol by Illumina at Helmholtz Zentrum, Munich, Germany. Samples were applied to the arrays in a randomized order. Aliquots of 1 μg of DNA were subjected to bisulphite conversion, and a 4-μl aliquot of bisulphite-converted DNA was subjected to whole-genome amplification, followed by enzymatic fragmentation and hybridization onto an Illumina Infinium MethylationEPIC BeadChip. The arrays were scanned with the iScan reader (Illumina). All analyzed samples have a sum of detection p-values across all the probes of less than 0.01. The logged (log_2) median of the methylated and unmethylated intensities of the analyzed samples clustered well visually. Furthermore, samples for which the actual sex did not match the predicted sex were excluded. Background subtraction and dye-bias normalization were performed via the noob method (Triche et al., 2013), followed by stratified quantile normalization. Probes with a detection p-value of more than 0.01 in 99% of the samples were filtered out. All pre-processing steps were performed using functions implemented in the minfi R/Bioconductor package (Aryee et al., 2014).

— S3 —

2 Design optimization

The study design as defined in the paper is a *mixOmics* specificity governing the associations between omics blocks in covariance maximization in the sparse multi-block partial least square (sMBPLS) framework. We used 20 singleton participants, without a twin among the 330 enrolled participants, to tune these associations between omics (where $c_{q,k}$ is the association between block q and k) and evaluate the impact on the quality of predictions.

2.1 Methodology

The choice of design also has an influence on the modeling as the design governs the links between omics blocks. The design matrix C is in our context a 4×4 matrix. The objective was to tune the design matrix to gauge the consequences of increasing or decreasing an association between blocks i.e., by tuning the $c_{q,k}$ elements between 0 and 1 in a direction that is detailed below. By construction, the design matrix C is symmetrical in the sense that $c_{q,k} = c_{k,q}\ \forall q, k$ which clearly reduces the calculation time.

Models with different designs were trained on the 310 participants (155 twin pairs) before predicting the systolic and diastolic blood pressure of the 20 singleton participants. These 20 singletons were involved in the variable selection stages, particularly in the transcriptomic and methylation data; their use for predictive purposes under different designs is therefore solely in the interest of detecting the best associations between blocks to minimise prediction error. In order to estimate the error committed on the prediction, a metric has been derived and based on the root mean square error (RMSE). The RMSE for SBP and DBP measurement m, noted $RMSE_m$ as defined in the (Δ_1) equation, was intended to estimate the error between SBP and DBP measurements $\hat{m_i}$ and their true measurements m_i for all individuals $i = 1, \ldots, N_s$ ($N_s = 20$), by squaring the prediction differences of every individuals.

$$\text{RMSE}_m = \left[\sum_{i=1}^{N} (m_i - \hat{m_i})^2 / N_s \right]^{1/2} \qquad (\Delta_1)$$

Because of the different orders of magnitude between systolic and diastolic pressure, the systolic RMSE was expected to fatally be stronger than the diastolic RMSE. In order to use only one criterion to minimize, a weighted averaged RMSE noted $RMSE_{adj}$ was preferred. This assigns a weight to each diastolic and systolic RMSE to obtain a weighted average of the two RMSEs with a greater penalty for deviations in predictions related to systolic blood pressure, as explained in equation (Δ_2).

$$RMSE_{adj} = \omega RMSE_{dia} + (1 - \omega) RMSE_{sys} \qquad (\Delta_2)$$

$$\text{with } \omega = \frac{1}{card(\mathcal{D})} \sum_{d_i \in \mathcal{D}} \left[\frac{RMSE_{sys,d_i}}{RMSE_{sys,d_i} + RMSE_{dia,d_i}} \right]$$

The term ω denotes the average importance over all tested designs $d_i \in \mathcal{D}$ of the systolic versus the diastolic RMSE. It has been estimated to be about 60 % (section 2.2). A design A was therefore considered better than a design B if the $RMSE_{adj}$ associated with the predictions from design A were lower than the RMSE from the predictions from design B.

2.2 Results

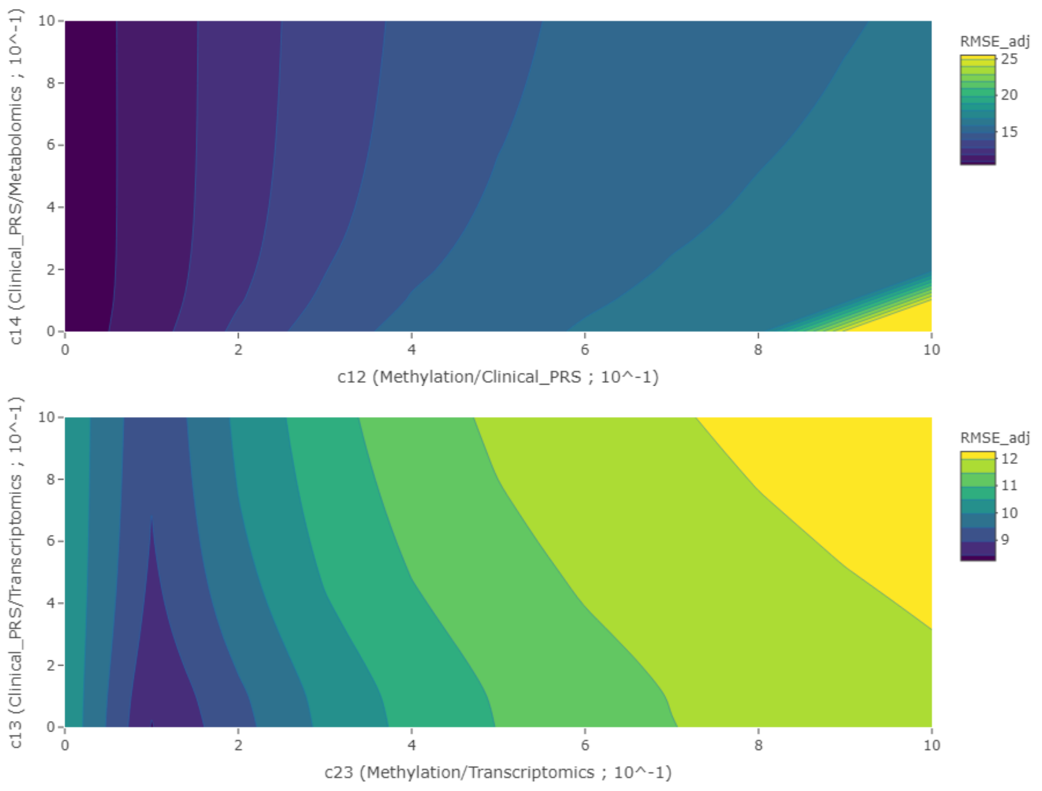
Before tuning the design matrix C, the cross-validation procedure (original paper ; supplementary document **S4**) was initiated to find out the optimal number of components needed to build the multi-omics associations. The goal was therefore to define upstream the number of components before investigating the possible associations between omics blocks. A large number of designs were then tested and the optimal number of components in the sense of the cross-validation procedure was always found to be far equal to 1. Supplemental Table 1 shows two examples obtained under two trivial designs noted null design (all associations between omics blocks are null i.e. $c_{qk} = 0$ for all q and k) and complete design (all associations are maximal i.e. $c_{qk} = 1$ for all q and k); k had therefore been fixed to 1.

Supplemental Table 1: CV score obtained by cross-validation on 3 maximal components for the two trivial null and complete designs

Number of components	Trivial design	pooled CV score	sd (20 iterations)
$k = 1$	Complete	164257.4	454.9
$k = 2$	Complete	311423.8	651.0
$k = 3$	Complete	541316.2	44879.1
$k = 1$	Null	168184.3	359.9
$k = 2$	Null	308303.7	748.1
$k = 3$	Null	299244.7	3508.7

whether in a null configuration (no association between omics) or a complete configuration (absolute associations between omics), a single component minimized the pooled CV score obtained by averaging the CV scores over the 20 iterations; the standard deviation (sd) was also relatively low, confirming the choice of a single component.

Concerning the associations between omics blocks governed by the design matrix C, due to the high number of operations to be performed, it was decided to perform a first step of observation between omics pairs in order to detect aberrant associations largely increasing the adjusted RMSE. Each c_{qk} ($\forall q, k \in \{1, 2, 3, 4\}$, $q \neq k$) coefficient was tuned two by two as partially shown in Supplemental Figure 1. The methylation data proved to be particularly unstable in that tuning the association between the methylation block and the clinical_PRS block (c_{12}) to non-zero values caused the adjusted RMSE to increase significantly.



The two graphs represent two pairs of tuned coefficients. The first one (above) shows how the association of the Methylation block with the Clinical_PRS block blows up the adjusted RMSE compared to the c_{14} coefficient that have a very small impact on the adjusted RMSE. The second plot (below) shows that a fairly weak association between transcriptome and methylome minimises the adjusted RMSE, with a much more reasonable scale than that found in the other contour plot.

Supplemental Figure 1: Contour plot of some combinations of omics associations and their impact on the $RMSE_{adj}$ measure.

A similar finding was reported in tuning the association between the Methylation block and the Metabolomics block (c_{24}). This is notably illustrated in Supplemental Figure 1 where increasing (c_{12}) significantly disturb the adjusted RMSE; a maximum deviation of the adjusted RMSE of more than 70% in both cases was observed when tuning the values of the c_{12} or c_{24} coefficients. Only the Transcriptomics-Methylation association was found to be relevant for a low c_{23} value ($c_{23} \simeq 0.1$) (Supplemental Figure 1). Paradoxically, certain combinations of block omics had little or no impact on the evolution of the adjusted RMSE, as in the case of the Clinical_PRS/Metabolomics combination (Supplemental Figure 1, coefficient c_{14}).

Once this first exploratory phase was completed, four main associations were submitted to the algorithm in order to determine those that minimized the adjusted RMSE: c_{13}, c_{14}, c_{23}, c_{34} (*Clinical_PRS:1 ;Methylation:2 ;Transcriptomics:3 ;Metabolomics:4*). The Methylation block was therefore only prospected to be coupled with the transcriptomic block, thus reducing exponentially the computation time. 11 probable values of each coefficient were prospected (from 0 to 1 by 0.1), leading to $M = 11^4 = 14'641$ adjusted RMSE calculated ($\omega = 60.0\%$). Two non-zero association coefficients minimized the adjusted RMSE; $c_{14} = 0.4$ and $c_{23} = 0.1$. However, the c_{14} coefficient has been particularly insensitive, as a change in its value had little impact on the adjusted RMSE (within one standard deviation).

It was finally observed that linking methylation data with transcriptomic data with a weak strength ($c_{23} = 0.1$) as well as clinical_PRS data with metabolomic data ($c_{14} = 0.4$) minimized the adjusted RMSE error in predicting the blood pressure values of the 20 singletons.

— S4 —

Integration of a cross validation procedure to the multi-block sparse partial least square (sMBPLS)

last version: January 2022

Forewords

In this Markdown document, we present a cross-validation method introduced in the paper by *Li et al.* (Li et al., 2012) and implemented using the features of *mixOmics (Rohart et al., 2017)*. This cross-validation method tunes the sparse arguments as well as the number of components to be selected in a multi-block sparse partial least squares (sMBPLS) framework. The implementation was done with R [R version 4.0.5] using the *block.spls mixOmics* function and all the features that revolve around it. **A complete tutorial applied to Opensource data is available upon request to the authors.**

1. Implementation of the cross-validation procedure

In the following, we use the same notation as in the article by *Li et al* (Li et al., 2012)(section 2.5, page 2461). The method described in this article is limited to 4 blocks to facilitate the understanding: the implementation carried out takes into account n blocks. The use of functions implemented in R base was preferred, and only the graphic representations relies on external aesthetic packages.

```
suppressWarnings(suppressMessages(library(mixOmics)))
# mixOmics is needed for the whole implementation
suppressWarnings(suppressMessages(library(ggplot2))) #graphics with ggplot2
suppressWarnings(suppressMessages(library(viridis)))
# viridis is used to take color palettes (optionnal : graphics)
suppressWarnings(suppressMessages(library(cowplot)))# using grids / ggplot2
```

Implementation limitation criteria:
The implemented method does not allow to select variables within the $\mathbf{Y}$ matrix: no sparse argument (λ_n) can be tuned. This argument λ_n has therefore been set to *M=ncol(Y)* and all the variables of the $\mathbf{Y}$ matrix have been kept. Only the arguments related to the blocks used to regress $\mathbf{Y}$ can be tuned. A design is also necessary to use the implemented cross-validation and must be defined beforehand, as shown subsequently. We have considered by default an identity matrix design of dimensions equal to the number of blocks plus one (*n+1*) but this design can be modified. Finally, the cross-validation is based on a distribution of individuals in L groups of equal size. The use of cross-validation can therefore only take place if the number of individuals in the study is the dividend of a Euclidean division with quotient L and remainder zero.

1.1 Inventory of implemented functions

The implementation of cross-validation has been segmented into several functions in order to facilitate both the understanding of the code and the step-by-step control of what has been coded. Among the five functions presented, one has a purely graphical purpose while the *cv.block.spls* function is the mother function using the last three purely technical functions. The role of each of these five functions is described below and the implementation details for each follow in subsections 1.2 through 1.6 .

◇ **put_indiv_in_L_groups** : This first technical function has the objective of distributing the individuals considered in the study in L groups. It corresponds to the first step *1)* of the *Li et al.* publication.

◇ **get_coefficients** : This second technical function takes as input the output of the *put_indiv_in_L_groups* function and applies part *2a)* of the *Li et al.* paper. It stores in output the projection coefficients defined as $\xi_i^l = X_i^i \mathbf{w}_i^{-l}$ $(i = 1, \dots, n-1)$ and $\zeta^l = Y^l \mathbf{q}^{-l}$ times the transposed associated loading matrices. The output is thus according to *Li et al.* notations a list of matrices corresponding to $\xi_i^l \mathbf{w}_i^{-l^T}$ and $\zeta^l \mathbf{q}^{-l^T}$ for each omics block i and each $l = 1, \dots, L$.

◇ **get_CVscore** : This technical function calculates the CV score defined in part *2b)* of the original publication. It takes as input the terms $\xi_i^l \mathbf{w}_i^{-l^T}$ and $\zeta^l \mathbf{q}^{-l^T}$ computed by the *get_coefficients* function. This function returns as output the CV score obtained for a defined combination of the parameters.

◇ **cv.block.spls** : The *cv.block.spls* function compiles all the outputs of the three technical functions. It mainly integrates the loops running these functions and at the end generates as output a list containing the CV scores according to all the combinations of the parameters, and this for each iteration.

◇ **plot.best.designs** : This function allows the user to get a graphical representation of the cross-validation CV scores when a number strictly greater than 2 omics are integrated as predictors. Because of the large combinations between parameters to be tested, it was chosen to only represent the best combination of parameters minimizing the CV score, by integrating error bars of length 1 standard deviation built with successive iterations.

1.2 function *put_indiv_in_L_groups*

The function *put_indiv_into_L_groups* has the unique goal of distributing all the individuals considered in L equal groups. It therefore takes as input both a list X composed of omics blocks, the number L of groups to be formed (L-fold cross validation) and a seed argument. This last argument seed has for simple objective to distribute the individuals in a unique way at each iteration of the cross-validation.

```
# L : number of groups to make
# current.seed is determined in the main mother function cv.block.spls
# X is only used to get the number of individuals according to the first block

put_indiv_in_L_groups <- function(L,current.seed,X){
  L <- L #optional
  set.seed(current.seed)
  myIDs <- rownames(X[[1]]) #IDs of all the individuals

  # give error message if nrow(X[[1]])%%L != 0  :
  if(nrow(X[[1]])%%L != 0){
    stop("Please choose L so that it distributes the number of individuals
         into groups of equal size ")
  }
  # The Li matrix will stock all individuals into their L groups
  Li <- matrix(data=NA,ncol=L,nrow=nrow(X[[1]])/L)

  for(i in 1:L){
```

```r
    Li[,i] <- sample(myIDs,replace = F,size = nrow(X[[1]])/L) #no replacement
    myIDs <- myIDs[-which(myIDs %in% Li[,i])]
  }
  return(Li) # final matrix with L columns and X[[1]])/L rows (individuals names)
}
```

1.3 function *get_coefficients*

The *get_coefficients* function is first used to obtain the ξ_i and ζ coefficients as defined in phase *2)* of the *Li et al.* paper. In output, it returns the terms $\xi_i^l \mathbf{w}_i^{-l^T}$ and $\zeta^l \mathbf{q}^{-l^T}$. It takes as input both the omics blocks $\mathbf{X}$ and the matrix $\mathbf{Y}$ to be regressed. It also depends on the function *put_indiv_into_L_groups* from which it inherits the matrix Li distributing the individuals in L equal groups. Finally, the function takes as input the number of components k to be retained when using the function *block.spls* as well as the argument *list.keepX* which depends directly on the combinations of the choosen sparse parameters.

```r
get_coefficients <- function(X,Y,list.keepX,k,Li){
  #build lists to store coefficients * t(loading matrices)
  CV.list <- list() # CV.list will store as many lists as omic blocks
  for(i in 1:length(X)){
    CV.list[[i]]<-list()
    names(CV.list)[i]<-paste0("CV",i)
  }
  # ad a list for zeta (Y matrix)
  CV.list[[length(X)+1]]<-list();names(CV.list)[length(X)+1]<-"CVzeta"

  ### cross-validation
  for(i in 1:ncol(Li)){ # for each group of individual l=1,...,L

    # preparing cross-validation datasets for group i
    X.cv <- list() #X for CV : X1^-l, ..., Xn-1^-l
    for(j in 1:length(X)){
      X.cv[[j]] <- X[[j]][-which(rownames(X[[j]]) %in% Li[,i]),]
    }
    names(X.cv)<-names(X) #get the same names to use block.spls function
    Y.cv <- Y[-which(rownames(Y)%in%Li[,i]),] # Y for CV : Y^-l
    names(Y.cv)<-names(Y)

    # tune the model with the X.cv and Y.cv datasets + list.keepX and k
    MyResult.diablo <- block.spls(X.cv, Y.cv, keepX=list.keepX,ncomp=k,
                                  design = MyDesign ,mode = "regression")
    # a design has been choosen upstream

    #store Xi coefficients for each omic layer
    Xi.list<-list() #this list only permits to store temporary the coefficients

    for(s in 1:length(X)){
      Xi.list[[s]] <- as.matrix(X[[s]][Li[,i],]) %*%
        as.matrix(MyResult.diablo$loadings[[s]])
      names(Xi.list)[s] <- paste0("Xi",s)
    }
    # store Zeta coefficients into Zeta Matrix
    Zeta <- as.matrix(Y[Li[,i],]) %*% as.matrix(MyResult.diablo$loadings$Y)
    # fill CV lists to get Xi^l * t(omega^l)
    for(s in 1:length(X)){
      CV.list[[s]][[i]] <- Xi.list[[s]] %*% t(as.matrix(MyResult.diablo$loadings[[s]]))
    }
    # fill the last CV argument to get Zeta^l * t(q^l)
    CV.list[[length(X)+1]][[i]] <- Zeta %*%
      t(as.matrix(MyResult.diablo$loadings[[length(X)+1]]))
  }
  # the CV.list is composed of n lists where n is the number of omics (+ Y)
  # for each of these lists, there are L matrices stored so that this matrices
  # are Xi^l * t(omega^l) (l=1,...,L) and Zeta^l * t(q^l) (l=1,...,L)
  return(CV.list)
}
```

1.4 function *get_CVscore*

The function *get_CVscore* takes several arguments insofar as it aims to give for each combination of sparse arguments λ_is (comb) a CV score defined in step *2b)* of *Li et al.* and that it stores in a dataframe *mydf* at row number *ind*. It inherits the coefficients multiplied to the transpose of the associated loading matrices (*CV.list*) as well as the *Li* matrix distributing the individuals into one of the L groups. The argument *combinations* is the matrix of the set of sparse argument combinations (λ_is). Finally, the list of blocks **X**, the matrix **Y** and the number of selected components *k* are filled in with the list of sparse arguments *list.keepX.cv*.

```
get_CVscore <- function(CV.list = CV.,Li,X=X,Y=Y, L,list.keepX.cv,k,comb,
                        mydf=mydf,combinations=combinations,ind=ind){
  # compute CV score
  CVscore <- 0 #initially zero

  ### first part of the CV score :

  for(i in 1:length(X)){
    for(l in 1:L){
      # the Frobenius norm is taken between the difference of
      # CV.list matrices and Xi^l
      norm. <- base::norm(as.matrix(X[[i]][which(rownames(X[[i]]) %in% Li[,l]),]) -
                          as.matrix(eval(parse(text = paste("CV.list[[",i,"]][[",l,"]]",
                                                            sep = "")))),type = "F")
      # sum for every iteration by putting penalties due to L/arguments
      CVscore <- CVscore + (norm.**2)/(length(Li[,l])*list.keepX.cv[[i]][1])
    } # we take list.keepX.cv[[i]][1] since the same lambda i is applied
    # to all the components
  }




  ### second part of the CV score :

  for(l in 1:L){
    # Frobenius norm with Y and zeta*t(q^-l)
    norm. <- base::norm(as.matrix(Y[which(rownames(Y) %in% Li[,l]),]) -
                as.matrix(eval(parse(text = paste("CV.list[[",length(X)+1,"]][[",l,"]]",
                                                                sep = "")))),type = "F")
    # sum
    CVscore <- CVscore + (norm.**2)/(length(Li[,l])*ncol(Y)) #Y non sparse
  } # we divide by M=ncol(Y) since we consider no sparsity for Y
  # store it in mydf
  mydf[ind,]<-c(k,as.integer(combinations[comb,]),CVscore)
  # the row number ind is filled with the number of component kept, the sparse
  # arguments and final CVscore obtained
  return(mydf)
}
```

1.5 function *cv.block.spls*

The function *cv.block.spls* is the mother function taking as input the data **X** and **Y** as well as a *lambda.list* containing all the sparse arguments to be tested. For each omics i, a vector containing the set of sparse arguments λ_i to test is specified. The number of components k to be tested is then scanned (from 1 to k) and the number of iterations is provided by the user. It is recommended to set the number of iterations to at least 10 so that the graphical output (*plot.best.designs* function) is robust.

```
# X is the list of omics layers
# Y must be a matrix or dataframe to regress
# lambda.list is the sparsity argument and must be a list
# n.repeat is how many times the cross-validation must be repeated

cv.block.spls <- function(X,Y,k=3L,lambda.list,n.iter=1L,show_progress=T){
  # check assumption : lambda.list must be a list
  if(is.list(lambda.list)==F){stop("lambda.list argument is not a list")}

  ### calculate number of iterations
  # number of needed iterations for each component :
  numb <- n.iter
  for(l in 1:length(lambda.list)){
    numb <- numb*length(lambda.list[[l]])
  }


  # all combinations to test : cross-validation procedure
  combinations <- expand.grid(lambda.list) # these combinations
  # need to be tested for each component k, n.iter times

  # build a final list that will store the tables mydf for each iteration
  CV.iter.final.score <- list() #will be the output


  #iteration steps

  for(iter in 1:n.iter){
    set.seed(.Random.seed[1:n.iter][iter]) #generate random seeds
    #important for the cross-validation and especially for the function
    # put_indiv_into_L_groups

    # Progress bar settings :
```

```
    if(show_progress==T){pb <- txtProgressBar(min = 0, max = numb*k/n.iter,
                                              style = 3,width = 40,char = "=")
    #the progress bar will be restarted at each iteration
    cat(paste(" // iteration number :", iter, ""))}


    # final output will be a list of theses dataframes :
    mydf <- data.frame(matrix(NA,nrow=numb/n.iter*k,ncol=(length(X)+2)))
    # +2 because : number of components and CVscore
    colnames(mydf)<-c("number of comp",names(X),"CVscore")

    ind <- 1 # row index to fill in mydf
    # for each component :
    for(k. in 1:k){
    #for each combination of sparse arguments
      for(comb in 1:nrow(combinations)){
        list.keepX.cv <- list() # we suppose same number of select var for all comp
        for(i in 1:(ncol(combinations))){
          list.keepX.cv[[i]] <- rep(combinations[comb,i],k.)}
        names(list.keepX.cv)<-names(X) #list.keepX.cv is the cross-validation
        # list.keepX : it takes a combination of sparse arguments and apply
        # them to each component k.

        ###### using technical functions to get CV scores for each combination
        Li. <- put_indiv_in_L_groups(10,.Random.seed[1:n.iter][iter] ,X)
        CV. <- get_coefficients(X,Y,list.keepX = list.keepX.cv,
                                k=k.,Li = Li.)
        mydf <- get_CVscore(CV.list = CV.,Li=Li.,X,Y,L=10,list.keepX.cv = list.keepX.cv,
                  k=k.,comb=comb,mydf=mydf,
                  combinations=combinations,ind=ind)
        ind <- ind+1 # once the CV score is done, increase the row number to impute
        if(show_progress==T){setTxtProgressBar(pb, ind)} #progress bar step
    }
    }
    if(show_progress==T){
      close(pb) #close progress bar at each end of iter=1,...,n.iter
    }
    CV.iter.final.score[[iter]]<-mydf # store the mydf
  }
  # the final output is a list of size n.iter.
  # The "mydf" at place p (p=1,...,n.iter) is the CV procedure at the pth iteration
  return(CV.iter.final.score)
}
```

1.6 function *plot.best.designs*

The *plot_best_designs* function aims at showing the *nb* best designs, i.e. the *nb* best models minimizing the CV score. This function takes the output of the function *cv.block.spls* as a list.

```
# nb=8 by default, if nb>=10 then the plot wont order the designs well:
# please select less than 10 designs or modify the function
```

```r
plot.best.designs <- function(nb=8,my_cv_model){

  # build a dataset :
  pooled_df <- as.data.frame(matrix(NA,nrow=nrow(my_cv_model[[1]]),
                                    ncol=ncol(my_cv_model[[1]])))
  colnames(pooled_df) <- colnames(my_cv_model[[1]])

  # get the pooled CV scores
  pooled_CVscore <- 0
  for(i in 1:length(my_cv_model)){
    pooled_CVscore <- pooled_CVscore + my_cv_model[[i]]$CVscore}
  # get the mean :
  pooled_CVscore <- pooled_CVscore/length(my_cv_model)

  # fill the dataset with these outputs
  final_df_output <- as.data.frame(matrix(NA,ncol=ncol(my_cv_model[[1]])-1,nrow=nb))
  final_df_output[,1:(length(my_cv_model[[1]])-1)] <-
    my_cv_model[[1]][which(pooled_CVscore<=sort(pooled_CVscore)[nb]),
                     1:(length(my_cv_model[[1]])-1)]
  colnames(final_df_output)<- colnames(my_cv_model[[1]][,1:(length(my_cv_model[[1]])-1)])
  final_df_output$pooledCVscore <-
    pooled_CVscore[which(pooled_CVscore<=sort(pooled_CVscore)[nb])]

  # store all the CV scores to get the standard deviation at the end
  allCVscores <- matrix(NA,ncol=length(my_cv_model),nrow=nb)
  for(i in 1:length(my_cv_model)){
    allCVscores[,i] <-
      as.vector(my_cv_model[[i]]$CVscore[which(pooled_CVscore<=sort(pooled_CVscore)[nb])])
  }
  # store CV score stand deviations :
  final_df_output$CVscoresd
  for(k in 1:nrow(final_df_output)){
    final_df_output$CVscoresd[k] <- sd(allCVscores[k,])
  }

  # ad the number of composants associated with each pooled CV score
  final_df_output$`number of comp` <- as.factor(final_df_output$`number of comp`)
  # order the dataset to get the best polled CV scores first
  final_df_output <- final_df_output[order(final_df_output$pooledCVscore),]
  rownames(final_df_output)<-make.names(rep("Design",nb+1),unique=T)[-1]
  final_df_output$Design <- rownames(final_df_output) #name them
  final_df_output$lb <- final_df_output$pooledCVscore -
    final_df_output$CVscoresd #lower bound
  final_df_output$ub <- final_df_output$pooledCVscore +
    final_df_output$CVscoresd #upper bound

  # require ggplot and viridis
  plot.out <-  ggplot(final_df_output,aes(Design,pooledCVscore,col=`number of comp`))+
    geom_point(size=4)+ theme_bw() +
    theme(axis.text.x = element_text(color="black", size=10, angle=35,vjust = 0.5)) +
  scale_color_manual(values=viridis(n=length(table(final_df_output$`number of comp`)))) +
  geom_errorbar(aes(ymin = lb, ymax = ub), width = 0.2, linetype="solid", alpha=0.6,
                position = position_dodge(width = 0.5), color="darkred", size=1) +
    xlab("") + ggtitle("Top designs minimizing the CV score")

  # final plot and final dataset of the nb best designs:
  return(list(plot.out,final_df_output))
}
```

Session info

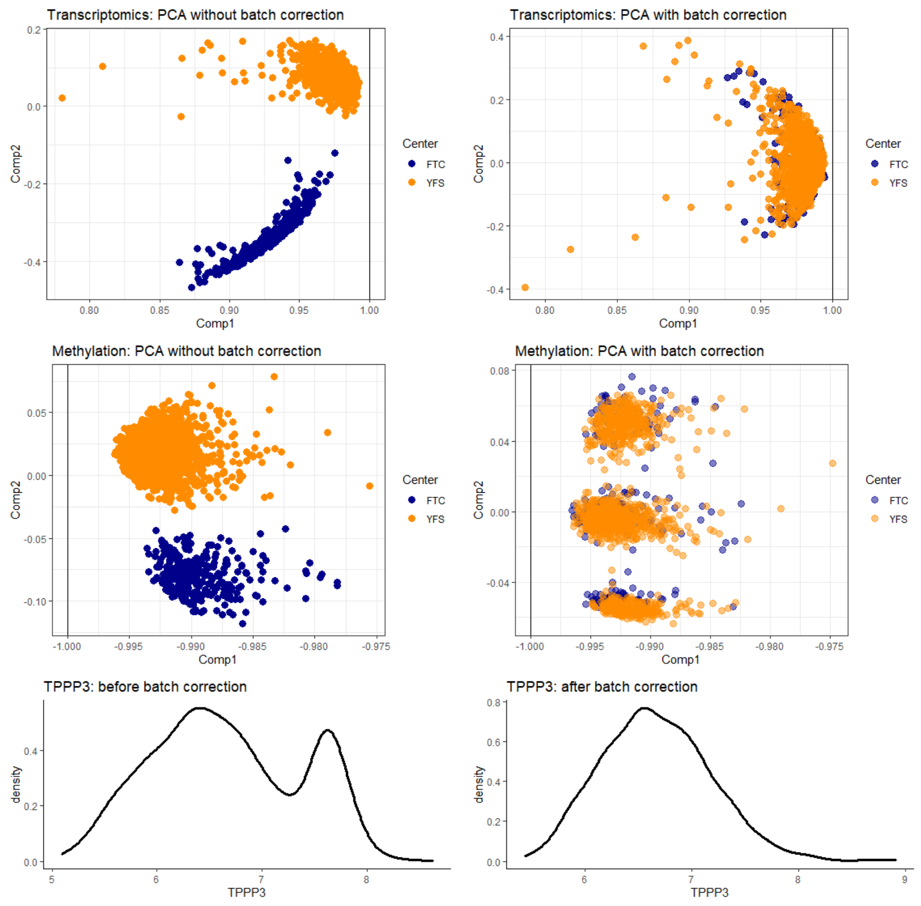
```r
sessionInfo()
```

```
## R version 4.0.5 (2021-03-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
## [3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
## [5] LC_TIME=French_France.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] cowplot_1.1.0     viridis_0.5.1     viridisLite_0.3.0 mixOmics_6.14.0
## [5] ggplot2_3.3.2     lattice_0.20-41   MASS_7.3-53
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.7         RSpectra_0.16-0    plyr_1.8.6         pillar_1.4.6
##  [5] compiler_4.0.5     RColorBrewer_1.1-2 tools_4.0.5        digest_0.6.26
##  [9] evaluate_0.14      lifecycle_0.2.0    tibble_3.0.4       gtable_0.3.0
## [13] pkgconfig_2.0.3    rlang_0.4.10       Matrix_1.3-2       igraph_1.2.6
## [17] ggrepel_0.8.2      yaml_2.2.1         parallel_4.0.5     xfun_0.18
## [21] gridExtra_2.3      withr_2.3.0        stringr_1.4.0      dplyr_1.0.2
## [25] knitr_1.30         generics_0.0.2     vctrs_0.3.4        grid_4.0.5
## [29] tidyselect_1.1.0   glue_1.4.2         ellipse_0.4.2      R6_2.4.1
## [33] rARPACK_0.11-0     rmarkdown_2.7      tidyr_1.1.2        reshape2_1.4.4
## [37] purrr_0.3.4        corpcor_1.6.9      magrittr_1.5       matrixStats_0.57.0
## [41] scales_1.1.1       ellipsis_0.3.1     htmltools_0.5.1.1  colorspace_1.4-1
## [45] stringi_1.5.3      munsell_0.5.0      crayon_1.3.4
```

— S5 —

Batch effect correction for Methylation and Transcriptomics blocks

Batch correction was performed (*sva* package) to merge FTC methylation and transcriptomic data with YFS data. The representation in Supplemental Figure 2 shows that the participants are arranged in the PCA reduced space according to the source cohort. The nature of these differences can't only be explained by clinical characteristics; non-biological factors are suspected (Supplemental Figure 2). The cohort effect was therefore corrected.

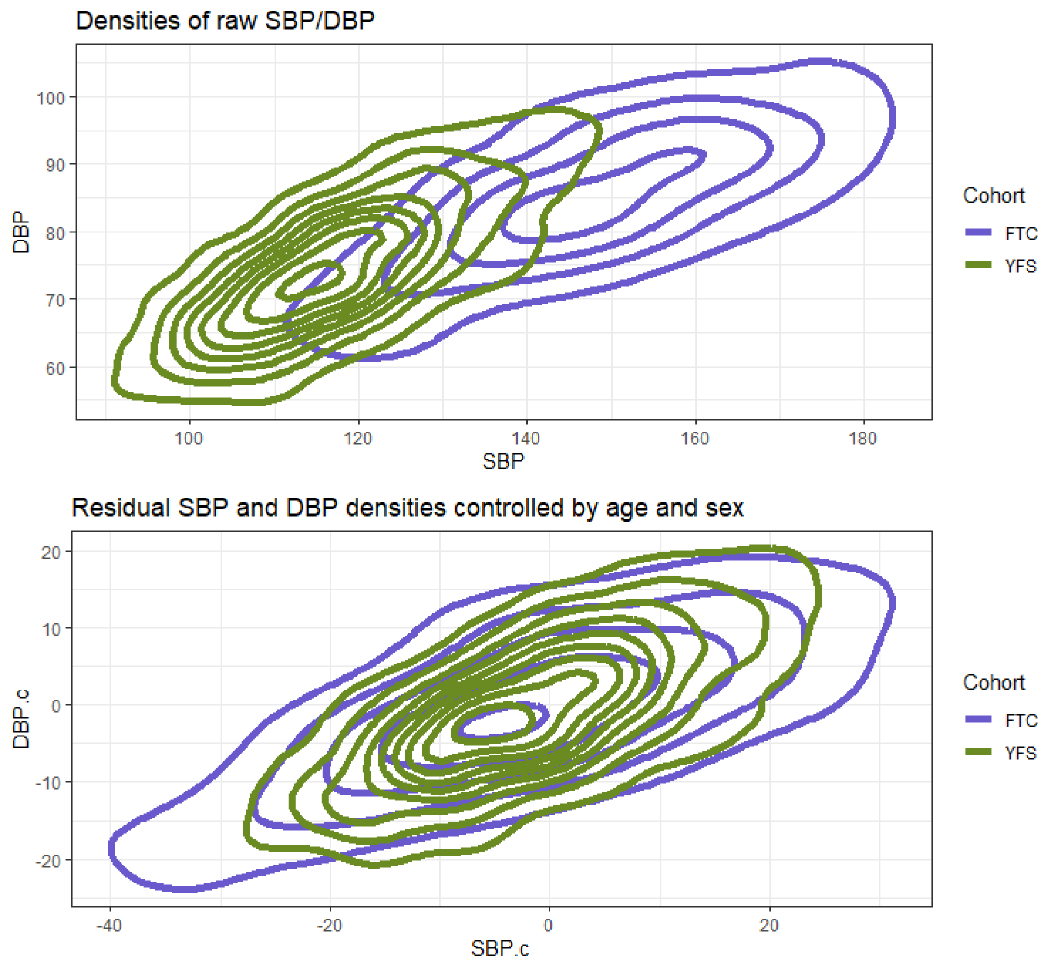


The top four plots show the projection of participants from both cohorts into the first plane obtained by PCA as part of the batch effect correction for transcriptomic and methylation data while the last two plots refer to the density of TPPP3 gene values with the two cohorts combined. Before batch correction, the *TPPP3* gene values are bi-distributed due to the "Cohort" effect.

Supplemental Figure 2: Correction of the *Cohort* effect for the Transcriptomic and Methylation blocks

— S6 —

The densities of SBP and DBP between the 2 study cohorts are substantially different (Supplemental Figure 3). SBP and DBP densities when controlled for age and sex were similar between the two cohorts YFS and FTC. Age and sex explain therefore a large part of the differences in SBP and DBP between the two cohorts.



Participants from the FTC Cohort tend to have higher SBP and DBP than participants from YFS (top). When controlled for age and sex, the blood pressure densities recenter and almost coincide; age and sex explain much of the inter-cohort differences in SBP and DBP.

Supplemental Figure 3: DBP and SBP densities for the training (FTC) and testing (YFS) cohort with and without controlling by age and sex.

— S7 —

Investigating the failure to integrate methylation data into modeling

The failure to integrate methylation data into the modeling raises many questions about the possible causes of this finding. As explained in the *Discussion* section of the original paper, pre-processing in an updated version was considered to investigate the impact of methylation pre-processing and variable selection on the blood pressure predictions obtained on the YFS cohort. 276 of the original 310 FTC participants were retained following the methodology used in van Dongen (van Dongen, 2021) as the methylation data for these participants was obtained in a different way than in the original paper (supplementary document, section S1). To also investigate possible problems in the variable selection by elastic-net when pre-processing the methylation data, 30 replicated CpG sites (Richard et al, 2017) were considered instead of the selection that was made in the original article. Single block models were explored first, i.e. only the methylation data were trained on the 276 FTC participants before predicting the blood measurements of the YFS participants. The coarse findings were as follows:

1. With new pre-processing and the same CpG variables as in the original study, the methylation data persisted in failing to predict SBP and DBP among YFS participants (p-value $> 5\%$ for both DBP and SBP, Spearman correlation nullity test).

2. With new pre-processing and 30 replicated CpG sites, the methylation data persisted in failing to predict systolic/diastolic pressure among YFS participants (p-value $> 5\%$ for both DBP and SBP, Spearman correlation nullity test).

3. With respect to point number 2); controlling for sex and age on blood measurements upstream in both cohorts did not improve predictions while age and sex explain most of the differences between the cohorts (supplementary document, section S6). However, controlling for sex, age and BMI upstream in the modeling yielded slight but still small improvements in predictions (Supplemental Table 2). The cost of these slight improvements lies in the control of gender, age and BMI. This is in fact limiting, as it would mean no more BMI measurements in the multi-omics modeling and no more relationships between metabolomic variables and BMI in the modeling.

4. Controlling blood measurements by age and/or sex and/or BMI upstream of the 3-block or 4-block modeling process would deteriate predictions. Better predictions were obtained when the age or BMI variables were kept in the model and not used as control variables upstream. In cases where sex, age and BMI were used upstream to regress systolic and diastolic blood pressures (and therefore removed from the modeling), a Spearman correlation coefficient significantly lower than 16% for SBP and DBP (higher bound, 95% confidence) is obtained for the 4-block (with methylation) and 3-block (without methylation) models.

It is not the pre-processing itself that seemed to be the cause of the integration failure of methylation data, and the upstream variable selection seemed to play a somewhat more significant role although this remains minor. To integrate methylation data into the modeling for predictive purposes, the cost is to control the blood measurements by several variables upstream (age, BMI, sex *etc.*) which to some extent goes against the multi-omics issues. Controlling for variables such as age or BMI allows to build a model that tends to explain what is not explained by these control variables, which does not work particularly well for predictive purposes.

The gist of multi-omics modeling is to capture relationships between variables of different natures. Controlling for age or BMI variables upstream of the modeling could be deleterious for the other omics blocks, insofar as the Clinical_PRS block would lose large predictors (and therefore the metabolomics block as well, as this block is linked to the Clinical_PRS block). As an example, the Clinical_PRS block is associated to the Metabolomics block by the BCAA-BMI relationships; the clinical variables Age and BMI actually act as safeguards within the model. Thus, one would lose the ability to interpret the model and determine relationships between variables. Integrating methylation data into multi-omics models to study complex phenotypes requires controls that could in some extent degrade both the predictive power of the model and the interpretation that can be drawn from it.

Supplemental Table 2: Spearman correlation coefficients between measured and predicted systolic and diastolic measurements in the YFS cohort

Blood pressure	Workflow	Spearman ρ	Confidence Interval (95%)
DBP	New workflow	0.064	[0.010 , 0.117]
SBP	New workflow	0.062	[0.008 , 0.115]
DBP	Original workflow	0.045	[-0.009 ; 0.098]
SBP	Original workflow	0.051	[-0.002 ; 0.104]

The differences between the two models in terms of Spearman's coefficient remain frail. The new workflow corresponds to the methylation data as pre-processed in van Dongen's paper (van Dongen et al., 2021), whereas the original workflow is the one used in the original paper.